

# Team Formation for Reformation in Multiagent Domains like RoboCupRescue

Ranjit Nair<sup>1</sup>, Milind Tambe<sup>1,2</sup>, and Stacy Marsella<sup>2</sup>

<sup>1</sup> Computer Science Department, University of Southern California  
Los Angeles CA 90089, USA  
{nair, tambe}@usc.edu

<sup>2</sup> University of Southern California's Information Sciences Institute  
Marina del Rey, CA 90292, USA  
marsella@isi.edu

**Abstract.** Team formation, i.e., allocating agents to roles within a team or subteams of a team, and the reorganization of a team upon team member failure or arrival of new tasks are critical aspects of teamwork. They are very important issues in RoboCupRescue where many tasks need to be done jointly. While empirical comparisons (e.g., in a competition setting as in RoboCup) are useful, we need a quantitative analysis beyond the competition — to understand the strengths and limitations of different approaches, and their tradeoffs as we scale up the domain or change domain properties. To this end, we need to provide complexity-optimality tradeoffs, which have been lacking not only in RoboCup but in the multiagent field in general.

To alleviate these difficulties, this paper presents *R-COM-MTDP*, a formal model based on decentralized communicating POMDPs, where agents explicitly take on and change roles to (re)form teams. R-COM-MTDP significantly extends an earlier COM-MTDP model, by introducing roles and local states to better model domains like RoboCupRescue where agents can take on different roles and each agent has a local state consisting of the objects in its vicinity. R-COM-MTDP tells us where the problem is highly intractable (NEXP-complete) and where it can be tractable (P-complete), and thus understand where algorithms may need to trade-off optimality and where they could strive for near optimal behaviors. R-COM-MTDP model could enable comparison of various team formation and reformation strategies — including the strategies used by our own teams that came in the top three in 2001 — in the RoboCup Rescue domain and beyond.

## 1 Introduction

The utility of the multi-agent team approach for coordination of distributed agents has been demonstrated in a number of large-scale systems for sensing and acting like disaster rescue simulation domains, such as RoboCupRescue Simulation Domain [5, 10] and sensor networks for real-time tracking of moving targets [7]. These domains contain tasks that can be performed only by collaborative

actions of the agents. Incomplete or incorrect knowledge owing to constrained sensing and uncertainty of the environment further motivate the need for these agents to explicitly work in teams. A key precursor to teamwork is team formation, the problem of how best to organize the agents into collaborating teams that perform the tasks that arise. For instance, in RoboCupRescue, injured civilians in a burning building may require teaming of two ambulances and three nearby fire-brigades to extinguish the fire and quickly rescue the civilians. If there are several such fires and injured civilians, the teams must be carefully formed to optimize performance.

Our work in team formation focuses on dynamic, multiagent environments, such as RoboCupRescue Simulation Domain [5, 10] and sensor networks [7]. In such domains teams must be formed rapidly so tasks are performed within given deadlines, and teams must be reformed in response to the dynamic appearance or disappearance of tasks. The problems with the current team formation work for such dynamic real-time domains are two-fold. First, most team formation algorithms [12, 4, 2, 3, 7] are static. In order to adapt to the changing environment the static algorithm would have to be run repeatedly.

Second, much of the work in RoboCupRescue has largely relied on experimental work and the competitions have been very useful in comparing various algorithms. A complementary technique is theoretical analysis. However, there has been a lack of theoretical analysis of algorithms, such as their worst-case complexity. This is especially important in understanding how algorithms work if domain parameters change, how they will scale up, etc.

In this paper we take initial steps to attack both these problems. As the tasks change and members of the team fail, the current team needs to evolve to handle the changes. In RoboCupRescue [5, 10], each re-organization of the team requires time (e.g., fire-brigades may need to drive to a new location) and is hence expensive because of the need for quick response. Clearly, the current configuration of agents is relevant to how quickly and well they can be re-organized in the future. Each re-organization of the teams should be such that the resulting team is effective at performing the existing tasks but also flexible enough to adapt to new scenarios quickly. We refer to this reorganization of the team as "Team Formation for Reformation". In order to solve the "Team Formation for Reformation" problem, we present *R-COM-MTDPs* (**R**oles and **C**ommunication in a **M**arkov **T**eam **D**ecision **P**rocess), a formal model based on communicating decentralized POMDPs, to address the above shortcomings. R-COM-MTDP significantly extends an earlier model called COM-MTDP [9], by making important additions of roles and agents' local states, to more closely model current complex multiagent teams. Thus, R-COM-MTDP provides decentralized optimal policies to take up and change roles in a team (planning ahead to minimize reorganization costs), and to execute such roles.

We use the disaster rescue domain to motivate the "Team Formation for Reformation" problem. We present real world scenarios where such an approach would be useful and use the RoboCup Rescue Simulation Environment [5, 10] to explain the working of our model. We show that the generation of optimal poli-

cies in R-COM-MTDPs is NEXP-complete although different communication and observability conditions significantly reduce such complexity. The nature of observability and communication in the RoboCupRescue domain makes it computationally intractable thus this motivating the study of optimality-complexity tradeoffs in approximation algorithms are necessary. R-COM-MTDPs provide a general tool for analysis of role-taking and role-executing policies in multiagent teams and for comparison of various approximate approaches. This will allow us to model the agents we developed for this domain [8] (which finished third at RoboCup 2001 and second at Robofesta, 2001) as an R-COM-MTDP and determine how changes in the agents' behavior could result in an improved performance. It is important that we develop tools in RoboCup that are relevant beyond RoboCup and contribute to the wider community. This work is aimed at meeting that objective.

## 2 Domain and Motivation

The RoboCupRescue Simulation Domain [5, 10], provides an environment where large-scale earthquakes can be simulated and heterogeneous agents can collaborate in the task of disaster mitigation. Currently, the environment is a simulation of an earthquake in the Nagata ward in Kobe, Japan. As a result of the quake many buildings collapse, civilians get trapped, roads get damaged and gas leaks cause fires which spread to neighboring buildings. There are different kinds of agents that participate in the task of disaster mitigation viz. fire brigades, ambulances, police forces, fire stations, ambulance centers and police stations. In addition to having a large number of heterogeneous agents, the state of the environment is rapidly changing – buried civilians die, fires spread to neighboring buildings, buildings get burnt down, rescue agents run out of stamina, etc. There is uncertainty in the system on account of incorrect information or information not reaching agents.

In such a hostile, uncertain and dynamically changing environment teams need to continually form and reform. We wish to understand the properties of such team formation and reformation algorithms. For instance, as new fires start up or fire engines get trapped under buried collapsing buildings, teams once formed may need to reform. While current algorithms in RoboCup Rescue for such reformation (and outside) react to such circumstances based solely on current information, in general, such reactive techniques may not perform well. In fact, methods that plan ahead taking future tasks and failures into account may be needed to minimize reformation, given that reformations take time. While we can perform useful empirical comparisons of current algorithms within the current competition settings, the field needs to analyze the performance of such algorithms for a wide variety of settings of agent failure rates, new task arrival rate, reformation costs, domain scale-up, and other factors. Theoretical analysis can aid in such analysis, providing us techniques to understand general properties of the algorithms proposed.

We focus in particular on a technique called "Team Formation for Reformation", i.e., teams formed with lookahead to minimize costs of reformation. The following real-world scenarios illustrate the need for such team formation for reformation.

1. A factory B catches fire at night. Since it is known that the factory is empty no casualties are likely. Without looking ahead at the possible outcomes of this fire, one would not give too much importance to this fire and might assign just one or two fire brigades to it. However, if by looking ahead, there is a high probability that the fire would spread to a nearby hospital, then more fire brigades and ambulances could be assigned to the factory and the surrounding area to reduce the response time. Moving fire brigades and ambulances to this area might leave other areas where new tasks could arise empty. Thus, other ambulances and fire brigades could be moved to strategic locations within these areas.
2. There are two neighborhoods, one with small wooden houses close together and the other with houses of more fire resistant material. Both these neighborhoods have a fire in each of them with the fire in the wooden neighborhood being smaller at this time. Without looking ahead to how these fires might spread, more fire brigades may be assigned to the larger fire. But the fire in the wooden neighborhood might soon get larger and may require more fire brigades. Since we are strapped for resources, the response time to get more fire brigades from the first neighborhood to the second would be long and possibly critical.
3. There is an unexplored region of the world from which no reports of any incident have come in. This could be because nothing untoward has happened in that region or more likely, considering that a major earthquake has just taken place, that there has been a communication breakdown in that area. By considering both possibilities, it might be best if police agents take on the role of exploration to discover new tasks and ambulances and fire brigades ready themselves to perform the new tasks that may be discovered.

Each of these scenarios demonstrate that looking ahead at what events may arise in the future is critical to knowing what teams will need to be formed. The time to form these future teams from the current teams could be greatly reduced if the current teams were formed keeping this future reformation in mind.

### **3 From COM-MTDP to R-COM-MTDP**

The COM-MTDP model[9] has two main advantages. First, COM-MTDP provides complexity analysis of team coordination given different communication assumptions. Even though it does not focus on team formation or reformation (which are topics of this paper), it serves as a basis for developing a new computational framework called R-COM-MTDP that provides such an analysis. Second, COM-MTDP was used to analyze different general team coordination algorithms based on the joint intentions theory, including the STEAM algorithm,

part of the ISIS teams that participated in RoboCup soccer [11]. COM-MTDP analysis revealed the types of domains where the team coordination behaved optimally, and specific domains where the algorithm communicated too much or too little — and the complexity of optimal communication strategies. Such analysis may in turn provide guidance in developing a new generation of team coordination algorithms which can intelligently engage in optimality-complexity tradeoffs. We attempt to do something similar with R-COM-MTDP.

### 3.1 COM-MTDP

Given a team of selfless agents,  $\alpha$ , a COM-MTDP [9] is a tuple,  $\langle S, A_\alpha, \Sigma_\alpha, P, \Omega_\alpha, O_\alpha, B_\alpha, R \rangle$ .  $S$  is a set of world states.  $A_\alpha = \prod_{i \in \alpha} A_i$  is a set of combined domain-level actions, where  $A_i$  is the set of actions available to agent  $i$ .  $\Sigma_\alpha = \prod_{i \in \alpha} \Sigma_i$  is a set of combined communicative actions, where  $\Sigma_i$  is the set of messages that agent  $i$  can broadcast to the other team members. The effects of domain-level actions obey the specified transition probability function,  $P(s_b, \mathbf{a}, s_e) = Pr(S^{t+1} = s_e | S^t = s_b, A_\alpha^t = \mathbf{a})$ .

$\Omega_\alpha = \prod_{i \in \alpha} \Omega_i$  is a set of combined observations, where  $\Omega_i$  is the set of observations that agent  $i$  may receive. The observation functions (or *information structures*),  $O_\alpha = \prod_{i \in \alpha} O_i$ , specify a probability distribution over observations that an agent may make, conditioned on the current state and combined actions of the agents:  $O_i(s, \mathbf{a}, \omega) = Pr(\Omega_i^t = \omega | S^t = s, A_\alpha^{t-1} = \mathbf{a})$ . We can define classes of information structures as in [9]:

**Collective Partial Observability:** We make no assumptions about the observability of the world state.

**Collective Observability:** There is a unique world state for the combined observations of the team:  $\forall \omega \in \Omega, \exists s \in S$  such that  $\forall s' \neq s, Pr(\Omega^t = \omega | S^t = s') = 0$ .

**Individual Observability:** Each individual’s observation uniquely determines the world state:  $\forall \omega \in \Omega_i, \exists s \in S$  such that  $\forall s' \neq s, Pr(\Omega_i^t = \omega | S^t = s') = 0$ .

In domains that are not individually observable, agent  $i$  chooses its actions and communication based on its belief state,  $b_i^t \in B_i$ , based on the observations and communication it has received through time  $t$ .  $B_\alpha = \prod_{i \in \alpha} B_i$  is the set of possible combined belief states. Agent  $i$  updates its belief state at time  $t$  when it receives its observation,  $\omega_i^t \in \Omega_i$ , and when it receives communication from its teammates,  $\Sigma_\alpha^t$ . We use separate state-estimator functions to update the belief states in each case: initial belief state,  $b_i^0 = SE_i^0()$ ; pre-communication belief state,  $b_{i \bullet \Sigma}^t = SE_{i \bullet \Sigma}(b_{i \Sigma \bullet}^{t-1}, \omega_i^t)$ ; and post-communication belief state,  $b_{i \Sigma \bullet}^t = SE_{i \Sigma \bullet}(b_{i \bullet \Sigma}^t, \Sigma_\alpha^t)$ .

Finally, the COM-MTDP reward function represents the team’s joint utility (shared by all members) over states, as well as both domain and communicative actions,  $R : S \times \Sigma_\alpha \times A_\alpha \rightarrow \mathbb{R}$ . We can express this overall reward as the sum of two rewards: a domain-action-level reward,  $R_A : S \times A_\alpha \rightarrow \mathbb{R}$ , and a communication-level reward,  $R_\Sigma : S \times \Sigma_\alpha \rightarrow \mathbb{R}$ . We can classify COM-MTDP

(and likewise R-COM-MTDP) domains according to the allowed communication and its reward:

**General Communication:** no assumptions on  $\Sigma_\alpha$  nor  $R_\Sigma$ .

**No Communication:**  $\Sigma_\alpha = \emptyset$ .

**Free Communication:**  $\forall \sigma \in \Sigma_\alpha, R_\Sigma(\sigma) = 0$ .

Analyzing the extreme cases, like free communication (and others in this paper) helps to understand the computational impact of the extremes. In addition, we can approximate some real-world domains with such assumptions.

### 3.2 R-COM-MTDP Extensions to COM-MTDP Model

We define a R-COM-MTDP as an extended tuple,  $\langle S, A_\alpha, \Sigma_\alpha, P, \Omega_\alpha, O_\alpha, B_\alpha, R, \mathcal{P}\mathcal{L} \rangle$ . The key extension over the COM-MTDP is the addition of subplans,  $\mathcal{P}\mathcal{L}$ , and the individual roles associated with those plans.

**Extension for Explicit Sub-Plans**  $\mathcal{P}\mathcal{L}$  is a set of all possible sub-plans that  $\alpha$  can perform. We express a sub-plan  $p_k \in \mathcal{P}\mathcal{L}$  as a tuple of roles  $\langle r_1, \dots, r_s \rangle$ .  $r_{jk}$  represents a *role instance* of role  $r_j$  for a plan  $p_k$  and requires some agent  $i \in \alpha$  to fulfill it. Roles enable better modeling of real systems, where each agent's role restricts its domain-level actions [13]. Agents' domain-level actions are now distinguished between two types:

**Role-Taking actions:**  $\mathcal{Y}_\alpha = \prod_{i \in \alpha} \mathcal{Y}_i$  is a set of combined role taking actions, where  $\mathcal{Y}_i = \{v_{ir_{jk}}\}$  contains the role-taking actions for agent  $i$ .  $v_{ir_{jk}} \in \mathcal{Y}_i$  means that agent  $i$  takes on the role  $r_j$  as part of plan  $p_k$ . An agent's role can be uniquely determined from its belief state.

**Role-Execution Actions:**  $\Phi_{ir_{jk}}$  is the set of agent  $i$ 's actions for executing role  $r_j$  for plan  $p_k$  [13].  $\Phi_i = \bigcup_{\forall r_{jk}} \Phi_{ir_{jk}}$ . This defines the set of combined execution actions  $\Phi_\alpha = \prod_{i \in \alpha} \Phi_i$ .

The distinction between role-taking and role-execution actions ( $A_\alpha = \mathcal{Y}_\alpha \cup \Phi_\alpha$ ) enables us to separate their costs. Within this model, we can represent the specialized behaviors associated with each role, and also any possible differences among the agents' capabilities for these roles. While filling a particular role,  $r_{jk}$ , agent  $i$  can perform only those role-execution actions,  $\phi \in \Phi_{ir_{jk}}$ , which may not contain all of its available actions in  $\Phi_i$ . Another agent  $\ell$  may have a different set of available actions,  $\Phi_{\ell r_{jk}}$ , allowing us to model the different methods by which agents  $i$  and  $\ell$  may fill role  $r_{jk}$ . These different methods can produce varied effects on the world state (as modeled by the transition probabilities,  $P$ ) and the team's utility (as modeled by the reward function,  $R_\Phi$ ). Thus, the policies must ensure that agents for each role have the capabilities that benefit the team the most.

In R-COM-MTDPs (as in COM-MTDPs), each decision epoch consists of two stages, a communication stage and an action stage. In each successive epoch,

the agents alternate between role-taking and role-execution epochs. Thus, the agents are in the role-taking epoch if the time index is divisible by 2, and are in the role execution epoch otherwise. Although, this sequencing of role-taking and role-execution epochs restricts different agents from running role-taking and role-execution actions in the same epoch, it is conceptually simple and synchronization is automatically enforced. As with COM-MTDP, the total reward is a sum of communication and action rewards, but the action reward is further separated into role-taking action vs. role-execution action:  $R_A(s, \mathbf{a}) = R_\gamma(s, \mathbf{a}) + R_\phi(s, \mathbf{a})$ . By definition,  $R_\gamma(s, \phi) = 0$  for all  $\phi \in \Phi_\alpha$ , and  $R_\phi(s, v) = 0$  for all  $v \in \gamma_\alpha$ . We view the role taking reward as the cost (negative reward) for taking up different roles in different teams. Such costs may represent preparation or training or traveling time for new members, e.g., if a sensor agent changes its role to join a new sub-team tracking a new target, there is a few seconds delay in tracking. However, change of roles may potentially provide significant future rewards.

We can define a role-taking policy,  $\pi_{i\gamma} : B_i \rightarrow \gamma_i$  for each agent’s role-taking action, a role-execution policy,  $\pi_{i\phi} : B_i \rightarrow \Phi_i$  for each agent’s role-execution action, and a communication policy  $\pi_{i\Sigma} : B_i \rightarrow \Sigma_i$  for each agent’s communication action. The goal is to come up with joint policies  $\pi_\gamma$ ,  $\pi_\phi$  and  $\pi_\Sigma$  that will maximize the total reward.

**Extension for Explicit Local States:  $S_i$**  In considering distinct roles within a team, it is useful to consider distinct subspaces of  $S$  relevant for each individual agent. If we consider the world state to be made up of orthogonal features (i.e.,  $S = \Xi_1 \times \Xi_2 \times \dots \times \Xi_n$ ), then we can identify the subset of features that agent  $i$  may observe. We denote this subset as its *local state*,  $S_i = \Xi_{k_{i1}} \times \Xi_{k_{i2}} \times \dots \times \Xi_{k_{im_i}}$ . By definition, the observation that agent  $i$  receives is independent of any features not covered by  $S_i$ :  $\Pr(\Omega_i^t = \omega | S^t = \langle \xi_1, \xi_2, \dots, \xi_n \rangle, A_\alpha^{t-1} = \mathbf{a}) = \Pr(\Omega_i^t = \omega | S_i^t = \langle \xi_{k_{i1}}, \dots, \xi_{k_{im_i}} \rangle, A_\alpha^{t-1} = \mathbf{a})$ .

**Role Decomposition** Some designers exploit roles further by decomposition into smaller subproblems, isolating the specific factors relevant to each of the separate roles [6, 15]. Within R-COM-MTDPs, role decomposition imposes two constraints on the agents’ role-taking and role-execution actions. First, we restrict the dynamics of the local state to depend on only the current local state and the agent’s domain-level action:  $\Pr(S_i^{t+1} | S^t = \langle \xi_1, \dots, \xi_n \rangle, A_\alpha^t = \prod_{j \in \alpha} a_j) = \Pr(S_i^{t+1} | S_i^t = \langle \xi_{k_{i1}}, \dots, \xi_{k_{im_i}} \rangle, A_i^t = a_i)$ . Second, we also structure the reward function so that the agents’ actions earn independent rewards:  $R(\langle \xi_1, \dots, \xi_n \rangle, \prod_{i \in \alpha} a_i) = \sum_{i \in \alpha} R_i(\langle \xi_{k_{i1}}, \dots, \xi_{k_{im_i}} \rangle, a_i)$ , where  $R_i$  is the local reward function, earned by agent  $i$ .

### 3.3 Application in RoboCupRescue

The notation described above can be applied easily to the RoboCup Rescue domain as follows:

1.  $\alpha$  consists of three types of agents: ambulances, police forces, fire brigades.
2. Injured civilians, buildings on fire and blocked roads can be grouped together to form tasks. The designer can choose how to form tasks, e.g. the world could be broken into fixed regions and all fires, hurt civilians and blocked roads within a region comprise a task. We specify sub-plans for each task type. These plans consist of roles that can be fulfilled by agents whose capabilities match those of the role.
3. We specify sub-plans,  $\mathcal{PL}$ , for each task type. Each sub-plan,  $p \in \mathcal{PL}$  comprises of a number of roles that need to be fulfilled by agents whose capabilities match those of the role in order to accomplish a task. For example, the task of rescuing a civilian from a burning building can be accomplished by a plan where fire-brigades first extinguish the fire, then ambulances free the buried civilian and one ambulance takes the civilian to a hospital. Each task can have multiple plans which represent multiple ways of achieving the task.
4. Each agent receives observations about the objects within its visible range. But there may be parts of the world that are not observable because there are no agents there. Thus, RoboCupRescue is a *collectively partially observable domain*. Therefore each agent, needs to maintain a belief state of what it believes the true world state is.
5. The reward function,  $R$  can be chosen to consider the capabilities of the agents to perform particular roles, e.g., police agents may be more adept at performing the “search” role than ambulances and fire-brigades. This would be reflected in a higher value for choosing a police agent to take on the “search” role than an ambulance or a fire-brigade. In addition, the reward function takes into consideration the number of civilians rescued, the number of fires put out and the health of agents.

The R-COM-MTDP model works as follows: Initially, the global world state is  $S^0$ , where each agent  $i \in \alpha$  has local state  $S_i^0$  and belief state  $b_i^0 = SE_i^0()$  and no role. Each agent  $i$  receives an observation,  $\omega_i^0$ , according to probability distribution  $O_i(S_i^0, null, \omega_i^0)$  (there are no actions yet) and updates its belief state,  $b_{i\bullet\Sigma}^0 = SE_{i\bullet\Sigma}(b_i^0, \omega_i^0)$  to incorporate this new evidence. In RoboCupRescue, each agent receives the complete world state before the earthquake as its first observation. Each agent then decides on what to broadcast based on its communication policy,  $\pi_{i\Sigma}$ , and updates its belief state according to  $b_{i\Sigma\bullet}^0 = SE_{i\Sigma\bullet}(b_{i\bullet\Sigma}^0, \Sigma_\alpha^0)$ . Each agent, based on its belief state then executes the role-taking action according to its role-taking policy,  $\pi_{i\Upsilon}$ . Thus, some police agents may decide on performing the “search role”, while others may decide to “clear roads”, fire-brigades decide on which fires “to put out”. By the central assumption of teamwork, all of the agents receive the same joint reward,  $R^0 = R(S^0, \Sigma_\alpha^0, A_\alpha^0)$ . The world then moves into a new state,  $S^1$ , according to the distribution,  $P(S^0, A_\alpha^0)$ . Each agent then receives the next observation about its new local state based on its position and its visual range and updates its belief state using  $b_{i\bullet\Sigma}^1 = SE_{i\bullet\Sigma}(b_{i\Sigma\bullet}^0, \omega_i^1)$ . This is followed by another communication action resulting in the belief state,  $b_{i\Sigma\bullet}^1 = SE_{i\Sigma\bullet}(b_{i\bullet\Sigma}^1, \Sigma_\alpha^1)$ . The agent then decides on a role-execution action based

on its policy  $\pi_{i\phi}$ . It then receives new observations about its local state and the cycle of observation, communication, role-taking action, observation, communication and role-execution action continues.

## 4 Complexity of R-COM-MTDPs

R-COM-MTDP supports a range of complexity analysis for generating optimal policies under different communication and observability conditions.

**Theorem 1.** *We can reduce a COM-MTDP to an equivalent R-COM-MTDP.*

*Proof.* Given a COM-MTDP,  $\langle S, A_\alpha, \Sigma_\alpha, P, \Omega_\alpha, O_\alpha, B_\alpha, R \rangle$ , we can generate an equivalent R-COM-MTDP,  $\langle S, A'_\alpha, \Sigma_\alpha, P', \Omega_\alpha, O_\alpha, B_\alpha, R' \rangle$ . Within the R-COM-MTDP actions,  $A'_\alpha$ , we define  $\Upsilon_\alpha = \{\text{null}\}$  and  $\Phi_\alpha = A_\alpha$ . In other words, all of the original COM-MTDP actions become role-execution actions in the R-COM-MTDP, where we add a single role-taking action that has no effect (i.e.,  $P'(s, \text{null}, s) = 1$ ). The new reward function borrows the same role-execution and communication-level components:  $R'_\Phi(s, \mathbf{a}) = R_A(s, \mathbf{a})$  and  $R'_\Sigma(s, \sigma)$ . We also add the new role-taking component:  $R'_\Upsilon(s, \text{null}) = 0$ . Thus, the only role-taking policy possible for this R-COM-MTDP is  $\pi'_{i\Upsilon}(b) = \text{null}$ , and any role-execution and communication policies ( $\pi'_\Phi$  and  $\pi'_\Sigma$ , respectively) will have an identical expected reward as the identical domain-level and communication policies ( $\pi_A$  and  $\pi_\Sigma$ , respectively) in the original COM-MTDP.  $\square$

**Theorem 2.** *We can reduce a R-COM-MTDP to an equivalent COM-MTDP.<sup>3</sup>*

*Proof.* Given a R-COM-MTDP,  $\langle S, A_\alpha, \Sigma_\alpha, P, \Omega_\alpha, O_\alpha, B_\alpha, R, \mathcal{P}\mathcal{L} \rangle$ , we can generate an equivalent COM-MTDP,  $\langle S', A_\alpha, \Sigma_\alpha, P', \Omega_\alpha, O_\alpha, B_\alpha, R' \rangle$ . The COM-MTDP state space,  $S'$ , includes all of the features,  $\Xi_i$ , in the original R-COM-MTDP state space,  $S = \Xi_1 \times \dots \times \Xi_n$ , as well as an additional feature,  $\Xi_{\text{phase}} = \{\text{taking, executing}\}$ . This new feature indicates whether the current state corresponds to a role-taking or -executing stage of the R-COM-MTDP. The new transition probability function,  $P'$ , augments the original function with an alternating behavior for this new feature:  $P'(\langle \xi_{1b}, \dots, \xi_{nb}, \text{taking} \rangle, v, \langle \xi_{1e}, \dots, \xi_{ne}, \text{executing} \rangle) = P(\langle \xi_{1b}, \dots, \xi_{nb} \rangle, v, \langle \xi_{1e}, \dots, \xi_{ne} \rangle)$  and  $P'(\langle \xi_{1b}, \dots, \xi_{nb}, \text{executing} \rangle, \phi, \langle \xi_{1e}, \dots, \xi_{ne}, \text{taking} \rangle) = P(\langle \xi_{1b}, \dots, \xi_{nb} \rangle, \phi, \langle \xi_{1e}, \dots, \xi_{ne} \rangle)$ . Within the COM-MTDP, we restrict the actions that agents can take in each stage by assigning illegal actions an excessively negative reward (denoted  $-r_{max}$ ):  $\forall v \in \Upsilon_\alpha$ ,  $R'_A(\langle \xi_{1b}, \dots, \xi_{nb}, \text{executing} \rangle, v) = -r_{max}$  and  $\forall \phi \in \Phi_\alpha$ ,  $R'_A(\langle \xi_{1b}, \dots, \xi_{nb}, \text{taking} \rangle, \phi) = -r_{max}$ . Thus, for a COM-MTDP domain-level policy,  $\pi'_A$ , we can extract role-taking and -executing policies,  $\pi_\Upsilon$  and  $\pi_\Phi$ , respectively, that generate identical behavior in the R-COM-MTDP when used in conjunction with identical communication-level policies,  $\pi_\Sigma = \pi'_\Sigma$ .  $\square$

<sup>3</sup> The proof of this theorem was contributed by Dr. David Pynadath

	Ind. Obs.	Coll. Obs.	Coll. Part. Obs.
No Comm.	P-Comp.	NEXP-Comp.	NEXP-Comp.
Gen. Comm.	P-Comp.	NEXP-Comp.	NEXP-Comp.
Free Comm.	P-Comp.	P-Comp.	PSPACE-Comp.

**Table 1.** Computational complexity of R-COM-MTDPs.

Thus, the problem of finding optimal policies for R-COM-MTDPs has the same complexity as the problem of finding optimal policies for COM-MTDPs. Table 1 shows the computational complexity results for various classes of R-COM-MTDP domains, where the results for individual, collective, and collective partial observability follow from COM-MTDPs [9] (Proof of COM-MTDP results are available at <http://www.isi.edu/teamcore/COM-MTDP/>). In the individual observability and collective observability under free communication cases, each agent knows exactly what the global state is. The P-Complete result is from a reduction from and to MDPs. The collectively partial observable case with free communication can be treated as a single agent POMDP, where the actions correspond to the joint actions of the R-COM-MTDP. The reduction from and to a single agent POMDP gives the PSPACE-Complete result. In the general case, by a reduction from and to decentralized POMDPs, the worst-case computational complexity of finding the optimal policy is NEXP-Complete.

Table 1 shows us that the task of finding the optimal policy is extremely hard, in general. However, reducing the cost of communication so it can be treated as if it were free has a potentially big payoff in complexity reduction. As can be seen from table 1, when communication changes from no communication to free communication, in collectively partially observable domains, the computational complexity changes from NEXP-Complete to PSPACE-Complete. In collectively observable domains, like the sensor domain, the computational savings are even greater, from NEXP-Complete to P-Complete. This emphasizes the importance of communication in reducing the worst case complexity. Table 1 suggests that if we were designers of a domain, we would increase the observability of the domain so as to reduce the computational complexity. However, in using existing domains, we don't have this freedom. The following section talks describes how R-COM-MTDPs could be useful in combating the computational complexity of "Team Formation for Reformation" in RoboCupRescue.

Table 2 shows the computational complexity of domains that are naturally role decomposable. Comparing tables 1 and 2, we see that role decomposition reduces the worst case complexity of R-COM-MTDPs from NEXP-Complete to PSPACE-Complete. In domains that are not naturally decomposable, we could explicitly treat roles such that there is role decomposition. This strategy would result in a policy that has less reward than the optimal but there will be considerable computational savings. Role decomposition further illustrates how R-COM-MTDP could be used to analyze different strategies and their complexity.

	Ind. Obs.	Coll. Obs.	Coll. Part. Obs.
No Comm.	P-Comp.	PSPACE-Comp.	PSPACE-Comp.
Gen. Comm.	P-Comp.	PSPACE-Comp.	PSPACE-Comp.
Free Comm.	P-Comp.	P-Comp.	PSPACE-Comp.

**Table 2.** Complexity of decomposable R-COM-MTDPs.

## 5 Analysis of RoboCupRescue

In RoboCupRescue, agents receive visual information of only the region in its surroundings. Thus no agent has complete knowledge of the global state of the world. Therefore the RoboCupRescue domain is in general *Collectively Partially Observable*. The number of communication messages that each agent can send or receive is restricted and in addition, communication takes up network bandwidth and so we cannot assume a communication policy where the agents communicate everything they see. Hence, RoboCupRescue comes under the *General Communication* case. Thus, the computational complexity of finding the optimal communication, role-taking and role-execution policies in RoboCupRescue is NEXP-Complete (see Table 1).

However the observability conditions are in our control— because we can devise agents that can try to provide collective observability. Thus what our results provide is guidance on “How to Design Teams” in Rescue, and what types of tradeoffs may be necessary based on the types of teams. What our results show is stunning – if we do collective observability and free communication then the complexity of our planning drops substantially. Now, as we know, this is not going to be possible — we can only approximate collective observability and free communication. However, we could then treat our result as an approximation – the resulting policy would be near-optimal but may be not optimal.

But, one of the biggest problems with developing agents for RoboCupRescue is not being able to compare different strategies easily. Owing to external factors like the packet loss and non-determinism internal to the simulation, a large number of simulations would be necessary to determine for certain if one strategy dominates another. However, just as in COM-MTDPs[9], where different approximate strategies for communication were analysed, RoboCupRescue can be modeled as an R-COM-MTDP. We could then treat alternative strategies as alternative policies and evaluate these. This would be very useful in making improvements to existing agents as well. For example, we could evaluate the policies specified by our agents[8], and evaluate their performance in the RoboCupRescue R-COM-MTDP. This would allow us to make changes to this policy and determine relatively quickly if this change resulted in an improvement. Such kind of incremental improvements could greatly improve the performance of our agents which finished in third place in RoboCup 2001 at Seattle and in second place in Robofesta 2001.

## 6 Summary and Related Work

This work addresses two shortcomings of the current work in team formation for dynamic multiagent domains: i) most algorithms are static in the sense that they don't anticipate for changes that will be required in the configuration of the teams, ii) complexity analysis of the problem is lacking. We addressed the first shortcoming by presenting *R-COM-MTDP*, a formal model based on decentralized communicating POMDPs, to determine the team configuration that takes into account how the team will have to be restructured in the future. *R-COM-MTDP* enables a rigorous analysis of complexity-optimality tradeoffs in team formation and reorganization approaches. The second shortcoming was addressed by presenting an analysis of the worst-case computational complexity of team formation and reformation under various types of communication.

While there are related multiagent models based on MDPs, they have focused on coordination after team formation on a subset of domain types we consider, and they do *not* address team formation and reformation. For instance, the *decentralized partially observable Markov decision process* (DEC-POMDP) [1] model focuses on generating decentralized policies in *collectively partially observable* domains with *no communication*; while the Xuan-Lesser model [14] focuses only on a subset of collectively observable environments.

Modi *et al.*[7] provide an initial complexity analysis of distributed sensor team formation, their analysis is limited to static environments (no reorganizations) — in fact, illustrating the need for R-COM-MTDP type analysis tools. Our complexity analysis illustrate where the problem is tractable and where it is not. Thus, telling us where algorithms could strive for optimality and where they should not. We intend to use the *R-COM-MTDP* model to compare the various team formation approaches including role decomposition for RoboCupRescue which were used by our agents in RoboCup-2001 and Robofesta 2001, where our agents finished in third place and second place respectively. It is important that tools be developed in RoboCup that step beyond RoboCup and contribute to the wider community. This work is a step in that direction.

## 7 Acknowledgment

We would like to thank David Pynadath for his discussions on extending the COM-MTDP model to R-COM-MTDP and Takayuki Ito for his invaluable contribution to the development of our Rescue agents, and the Intel Corporation for their generous gift that made this research possible.

## References

1. Bernstein, D. S., Zilberstein, S., Immerman, N.: The Complexity of Decentralized Control of MDPs. Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (2000)

2. Fatima, S. S., Wooldridge, M.: Adaptive Task and Resource Allocation in Multi-agent Systems. Proceedings of the Fifth International Conference on Autonomous Agents (2001)
3. Horling, B., Benyo, B., Lesser, V.: Using Self-Diagnosis to Adapt Organizational Structures. Proceedings of the Fifth International Conference on Autonomous Agents (2001)
4. Hunsberger, L., Grosz, B.: A Combinatorial Auction for Collaborative Planning. Proceedings of the Fourth International Conference on Multiagent Systems (2000)
5. Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjoh, A., Shimada, S.: RoboCupRescue: Search and Rescue for Large Scale Disasters as a Domain for Multi-agent Research. Proceedings of IEEE International Conference on Systems, Man and Cybernetics (1999)
6. Marsella, S., Adibi, J., Alonaizon, Y., Kaminka, G., Muslea, I., Tambe, M.: On being a teammate: Experiences acquired in the design of robocup teams. Proceedings of the Third International Conference on Autonomous Agents(1999)
7. Modi, P. J., Jung, H., Tambe, M., Shen, W.-M., Kulkarni, S.: A Dynamic Distributed Constraint Satisfaction Approach to Resource Allocation. Proceedings of Seventh International Conference on Principles and Practice of Constraint Programming (2001)
8. Nair,R., Ito, T., Tambe, M., Marsella, S.: Task Allocation in the Rescue Simulation Domain. RoboCup-2001: The Fifth Robot World Cup Games and Conferences, *Eds. Coradeschi, S., Tadokoro, S., Andreas Birk*, Springer-Verlag (2001)
9. Pynadath, D., Tambe, M.: Multiagent Teamwork: Analyzing the Optimality Complexity of Key Theories and Models. Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (2002)
10. Tadokoro, S., Kitano, H., Tomoichi, T., Noda, I., Matsubara, H., Shinjoh, A., Koto, T., Takeuchi, I., Takahashi, H., Matsuno, F., Hatayama, M., Nobe, J., Shimada, S.: The RoboCup-Rescue: An International Cooperative Research Project of Robotics and AI for the Disaster Mitigation Problem. Proceedings of SPIE 14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls (AeroSense), Conference on Unmanned Ground Vehicle Technology II (2000)
11. Tambe, M.: Towards Flexible Teamwork. Journal of Artificial Intelligence Research, **7**, 83-124 (1997)
12. Tidhar, G., Rao, A. S., Sonenberg, E.: Guided Team Selection. Proceedings of the Second International Conference on Multi-Agent Systems (1996)
13. Wooldridge, M., Jennings, N., Kinny, D.: A Methodology for Agent Oriented Analysis and Design. Proceedings of the Third International Conference on Autonomous Agents (1999)
14. Xuan, P., Lesser, V., Zilberstein, S.: Communication Decisions in Multiagent Cooperation. Proceedings of the Fifth International Conference on Autonomous Agents (2001)
15. Yoshikawa, T.: Decomposition of Dynamic Team Decision Problems. Proceedings of the IEEE, **AC-23**, 4, 627-632 (1978)